ITS assignment 4

Noah Jensen (xdr622) & Julian Pedersen (rsk975)

September 2022

SEED

Task 2

Task 2.A

In this subtask we had to change some configurations in the iptables on a router and then trying to access said router.

Rules

- iptables -A INPUT -p icmp –icmp-type echo-request -j ACCEPT
- iptables -A OUTPUT -p icmp –icmp-type echo-request -j ACCEPT
- iptables -P OUTPUT DROP
- iptables -P INPUT DROP

Rule 1 appends a rule to input which accepts pings from from the host to the router. Rule 2 appends a rule to output which accepts pongs (replys) to the pings from the router to the host. Rule 3 sets the protocol for output to drop every package by default (except those specified in other rules like rule 2). Rule 4 sets the protocol for input to drop every package by default (except those specified in other rules like rule 1) We set the firewall such that one can only

```
root@aabe68cb5269:/# ping seed-router
PING seed-router (10.9.0.11) 56(84) bytes of data.
64 bytes from seed-router.net-10.9.0.0 (10.9.0.11): icmp_seq=1 ttl=64 time=0.079 ms
64 bytes from seed-router.net-10.9.0.0 (10.9.0.11): icmp_seq=2 ttl=64 time=0.092 ms
--- seed-router ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9197ms
rtt min/avg/max/mdev = 0.067/0.076/0.092/0.007 ms
root@aabe68cb5269:/# telnet -l seed seed-router
Trying 10.9.0.11...
telnet: Unable to connect to remote host: Connection timed out
```

Figure 1: able to ping but not telnet

ping the router but not access it. This was tested using two methods. First we tried pinging the router from another docker, which was successful. Then we tried to telnet into the router, this however was not possible. That was to be expected as this what the job of our firewall on the router.

Task 2.B

In this task we had to follow some restrictions over the ICMP protocol that dictates traffic in and out of an interface. To satisfy the four restrictions we needed to make three rules in our iptables on the router.

Rules:

- iptables -A FORWARD -i eth1 -p icmp –icmp-type echo-request -j ACCEPT
- iptables -A FORWARD -i eth1 -p icmp –icmp-type echo-reply -j ACCEPT
- iptables -P FORWARD DROP

The first rule means that every echo-request (ping) that goes inside the interface (eth1) is to be accepted. Rule two means that every echo-reply (pong) that comes from inside the interface (eth1) is acceptable. The last rule dictates that all other traffic is not allowed. We testing if our three rules was sufficient to

```
root@5cfc3271ab42:/# ping 192.168.60.5
PING 192.168.60.5 (192.168.60.5) 56(84) bytes of data.
^C
--- 192.168.60.5 ping statistics ---
2 packets transmitted, 0 received, 100% packet loss, time 1004ms

root@5cfc3271ab42:/# ping seed-router
PING seed-router (10.9.0.11) 56(84) bytes of data.
64 bytes from seed-router.net-10.9.0.0 (10.9.0.11): icmp_seq=1 ttl=64 time=0.085 ms
64 bytes from seed-router.net-10.9.0.0 (10.9.0.11): icmp_seq=2 ttl=64 time=0.212 ms
64 bytes from seed-router.net-10.9.0.0 (10.9.0.11): icmp_seq=3 ttl=64 time=0.275 ms
64 bytes from seed-router.net-10.9.0.0 (10.9.0.11): icmp_seq=4 ttl=64 time=0.442 ms
64 bytes from seed-router.net-10.9.0.0 (10.9.0.11): icmp_seq=5 ttl=64 time=0.214 ms
^C
--- seed-router ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4093ms
rtt min/avg/max/mdev = 0.085/0.245/0.442/0.116 ms
```

Figure 2: able to ping router but not internal network

the restrictions by trying to ping both from the inside and outside the interface. As seen in the figure it is possible for the outside `10.9.0.5` to ping the router `10.9.0.11` but not the internal host `192.168.60.5`. In turn the internal host `192.168.60.5` can ping the external host `10.9.0.5`.

Task 2.C

In this task we had to follow some restrictions over a TCP connection that dictates traffic flow in and out of an interface. To satisfy the five given restriction we set up the three following rules.
Rules:

- iptables -A FORWARD -i eth0 -p tcp –destination 192.168.60.5 -j ACCEPT
- iptables -A FORWARD -i eth1 -p tcp –sport 23 -j ACCEPT
- iptables -P FORWARD DROP

The first rule allows packet forwarding of all packets inbound from an external network which have the destination 192.168.0.5, which is the IP address of the internal network machine Host 1. This allows for clients on an external network

```
root@68e14b374bac:/# telnet -l seed 192.168.60.5
Trying 192.168.60.5...
Connected to 192.168.60.5.
Escape character is '^]'.
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-54-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.
Last login: Sat Oct  8 21:14:29 UTC 2022 on pts/2
seed@5c554df1fca0:~$ echo bonjour
bonjour
root@68e14b374bac:/# telnet -l seed 192.168.60.6
Trying 192.168.60.6...
telnet: Unable to connect to remote host: Connection timed out
root@68e14b374bac:/# telnet -l seed 192.168.60.7
Trying 192.168.60.7...
telnet: Unable to connect to remote host: Connection timed out
```

Figure 3: External hosts can only connect to 192.168.60.5

to establish a connection to a single internal machine in the internal network with the corresponding IP address.

The second rule is similar, allowing for packet forwarding of all packets inbound to the router from the internal network which have the source port 23. Port 2 is the port which SSH is set to listen to, allowing for SSH connections between clients in the internal network as well as clients from an external network (as long as they pass the first rule).

The third rule is a policy change which drops all packet forwarding by default, meaning packets must clear one of the above two rules to pass.

Task 3

Task 3.A

In this task we had to experiment with connection tracking. The results is seen in the figure.

ICMP: When sending the ping to `192.168.60.5` we let it ping 3 times before killing the connection. Using the `conntrack` command at the router found that the connection was kept for 27 seconds. We could also see the `ICMP` types was 8, which means `echo-request` and 0, which means `echo-reply`.

UDP: when sending a `UDP` packet to the netcat server set up at IP `192.169.60.5` we gave the message *hej* and closed the connection. The `conntrack` command found that the connection was kept for 4 seconds on the router. We also saw

```
root@5c554df1fca0:/# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
12: eth0@if13: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:c0:a8:3c:05 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 192.168.60.5/24 brd 192.168.60.255 scope global eth0
       valid_lft forever preferred_lft forever
root@5c554df1fca0:/# telnet -l seed 10.9.0.5
Trying 10.9.0.5...
telnet: Unable to connect to remote host: Connection timed out
root@5c554df1fca0:/# telnet -l seed 192.168.60.6
Trying 192.168.60.6...
Connected to 192.168.60.6.
Escape character is '^]'.
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-54-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

This system has been minimized by removing packages and content that are
not required on a system that users do not log into.
```

Figure 4: Internal hosts can communicate with each other within the internal network.

```
root@d0d9c91e39c7:/# conntrack -L
tcp      6 109 TIME_WAIT src=10.9.0.5 dst=192.168.60.5 sport=40596 dport=9090 src=192.168.60.5 dst
=10.9.0.5 sport=9090 dport=40596 [ASSURED] mark=0 use=1
udp      17 4 src=10.9.0.5 dst=192.168.60.5 sport=36118 dport=9090 [UNREPLIED] src=192.168.60.5 ds
t=10.9.0.5 sport=9090 dport=36118 mark=0 use=1
conntrack v1.4.5 (conntrack-tools): 2 flow entries have been shown.
root@d0d9c91e39c7:/# conntrack -L
icmp     1 27 src=10.9.0.5 dst=192.168.60.5 type=8 code=0 id=38 src=192.168.60.5 dst=10.9.0.5 type
=0 code=0 id=38 mark=0 use=1
```

Figure 5: TCP, UDP and ICMP via conntrack

the packet was `[UNREPLIED]` which means no reply to the `UDP` packet.
TCP: when sending a `TCP` packet to the `netcat` server set up at IP `192.169.60.5` we gave the message *hej* and closed the connection. The `conntrack` command found that the connection was kept for 109 seconds on the router. Here the packet was `[ASSURED]` which means the connection is kept open to wait for the next packet. Lastly the `TCP` packet had the label `TIME_WAIT` which means it is waiting for a timeout to occur such that it can close the connection.
The reason why TCP keeps the connection for longer is because TCP starts with a 3-way handshake consisting of a SYN, SYN-ACK and ACK. After this the connection is kept open which is also seen in the assured label. To terminate a TCP connection the user has to send a FIN flag which the connection will wait for before timing out.

On the other hand a UDP connections does not need to establish an open connection between the sender and receiver, thus the packet will just be sent. This explains why the timer is low on the UDP connection.

ICMP connections have a default timeout at 30 seconds, this means this is how long the server will wait for a new packet.

Task 3.B

Rules:

- iptables -A FORWARD -p tcp -m conntrack –ctstate ESTABLISHED,RELATED -j ACCEPT
- iptables -A FORWARD -i eth0 -p tcp –destination 192.168.60.5 –dport 23 –syn -m conntrack –ctstate NEW -j ACCEPT
- iptables -A FORWARD -i eth1 -p tcp –dport 23 –syn -m conntrack –ctstate NEW -j ACCEPT
- iptables -P FORWARD DROP

All packages in an established connection is allowed according to rule 1.

Rule 2 forwards incoming SYN packets from the external network to the internal network, as long as they have a destination host 192.168.60.5 at port 23. In our network, this allows external hosts to establish a telnet-only connection to Host-1 in the internal network.

Rule 3 forwards all outgoing SYN packets from the internal network to the external network, as long as they are directed to port 23. This means that all internal hosts can establish a telnet-only connection with any external hosts.

The biggest different between task 2c and this task is that we can makes rules dependent on the type of package being sent and not just the protocol. The advantage of this is a more controllable firewall which contains more details, because you are able to analyse each connection, its context, and even the packet content itself, instead of just the packets in isolation. A stateless might be better suited for small upstarts who do not have much traffic yet and thus is not a big target either; stateless may also be better for networks with large amounts of simple traffic, as stateless firewalls do not process packets but simply check if a packet matches existing security rules. While a stateful offers better security thus is a must for establish companies.

Short answers

Question 1

What is a VPN?
A VPN is a virtual private network, that uses encryption over the public internet to help ensure sensitive data. It uses IPSec to give this encryption: the package has a IPv4 header that is detected by the router, while the payload is includes an IPsec header and encrypted datagram. A VPN is a cheaper and more realistic alternative to a real private network.

Question 2

Can a host firewall block malware from communicating from an infected computer back to its controller on the internet?
Yes. Firewalls have both inbound and outbound filters for this exact reason. An outbound filter can be configured to block communication with suspicious clients/servers on the internet to frustrate such malware attacks.

Question 3

Is an IPsec connection to your company network secure even if your home router's TCP/IP implementation has a buffer overflow vulnerability?
An IPSec datagram gets wrapped in a IPv4 header when going through the router, thus it becomes vulnerable to a buffer overflow attack like a normal IP connection. However, the datagram is encrypted so the attackers might not get much information out of it, but the attacker can still crash the user's computer or cause other damages.

Question 4

What is the difference between IPSec tunnel and transport mode?
Transport mode is host-to-host communication only and has end-to-end encryption. A transport-mode datagram has only the payload encrypted, meaning that the end host must decrypt to see the contents, providing end-to-end encryption.

Tunnel mode is either network-to-network or host-to-network and does not include end-to-end encryption and is able to do second-stage delivery within the virtual private network because of network gateways. A tunnel-mode datagram has the entire IP datagram encrypted; this means that after the ESP/AH headers are consumed by the gateways, the entire IP packet is unencrypted which is why tunnel mode does not provide end-to-end encryption.